



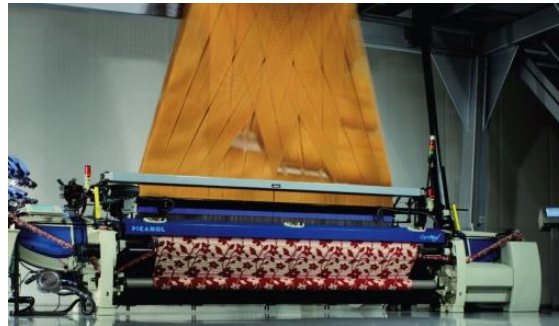
USING PAPYRUS IN A DESIGN SPACE EXPLORATION TOOLCHAIN

CURRENT DEVELOPMENTS AT FLANDERS MAKE

Who is Flanders Make?

A Flemish research institute whose mission is to strengthen the **long-term international competitiveness** of the Flemish manufacturing industry by carrying out **excellent, industry-driven, pre-competitive research** in the domains of

- ▲ **Mechatronics**
- ▲ **Product development methods**
- ▲ **Advanced manufacturing technologies**



Aiming at product & process innovation
for the **vehicles, machines** and **factories of the future**

Our partner network





Outline

▲ Use case

- ▲ Introduction
- ▲ Design space model
- ▲ Toolchain

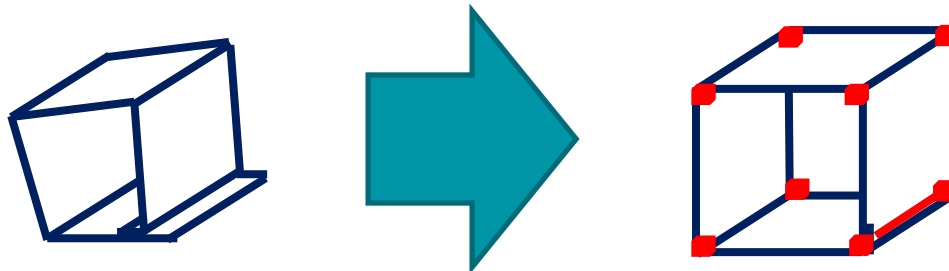
▲ Discussion

- ▲ Priorities
- ▲ Tool usability
- ▲ OCL framework
- ▲ Instance creation and visualization



Conceptual design of a robotic assembly cell

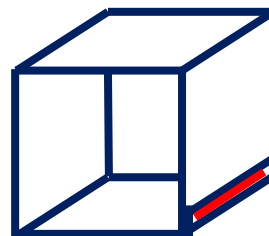
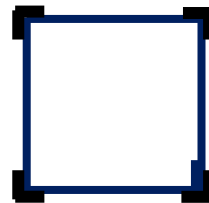
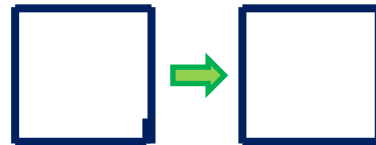
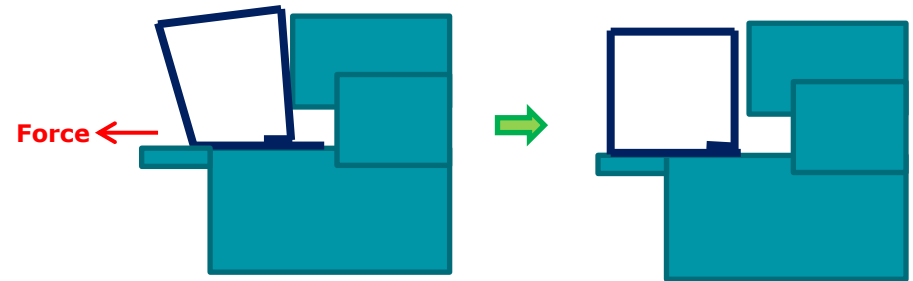
- ▲ Product to be manufactured
 - ▲ Casing for valve in ventilation system
- ▲ Requirements
 - ▲ Produce 40 cases per hour
 - ▲ Fully automated, no operator involved
 - ▲ Use existing machine repository
 - ▲ Design cell with minimal cost



Conceptual design of a robotic assembly cell

▲ Process steps

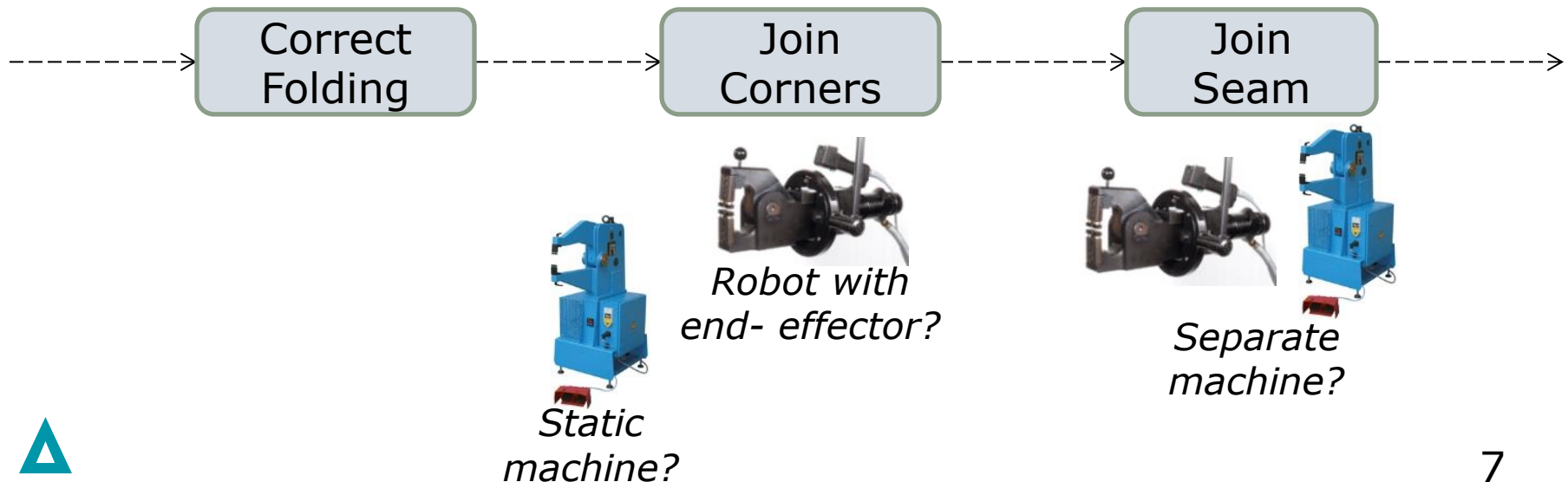
- ▲ Extract from bending machine
- ▲ Correct folding
- ▲ Join corners
- ▲ Join seam



Conceptual design of a robotic assembly cell



- ▲ What is the best way of producing this product?
 - ▲ What machines do we use per process step?
 - Different ways (“working principles”) to perform a process step
 - ▲ How are these machines assigned to the process steps?
 - Machines can be shared between steps (speed vs cost)



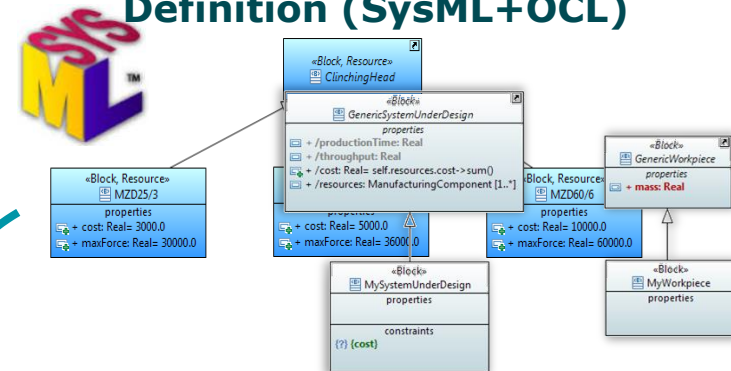
Conceptual design of a robotic assembly cell

▲ Approach: Computational design synthesis

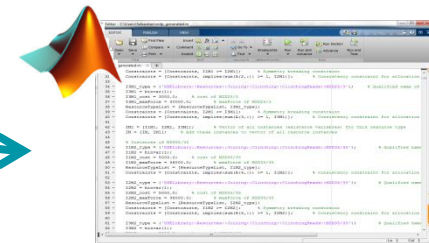
- ▲ Represent design problem in a formal model (SysML+OCL)
- ▲ Design repository to store knowledge for computational synthesis
- ▲ Automated transformation to a Mixed-Integer Linear Program (MILP)
- ▲ Represent solution in SysML

→ Explore more of the design space at a lower cost

Design Repository + Problem Definition (SysML+OCL)



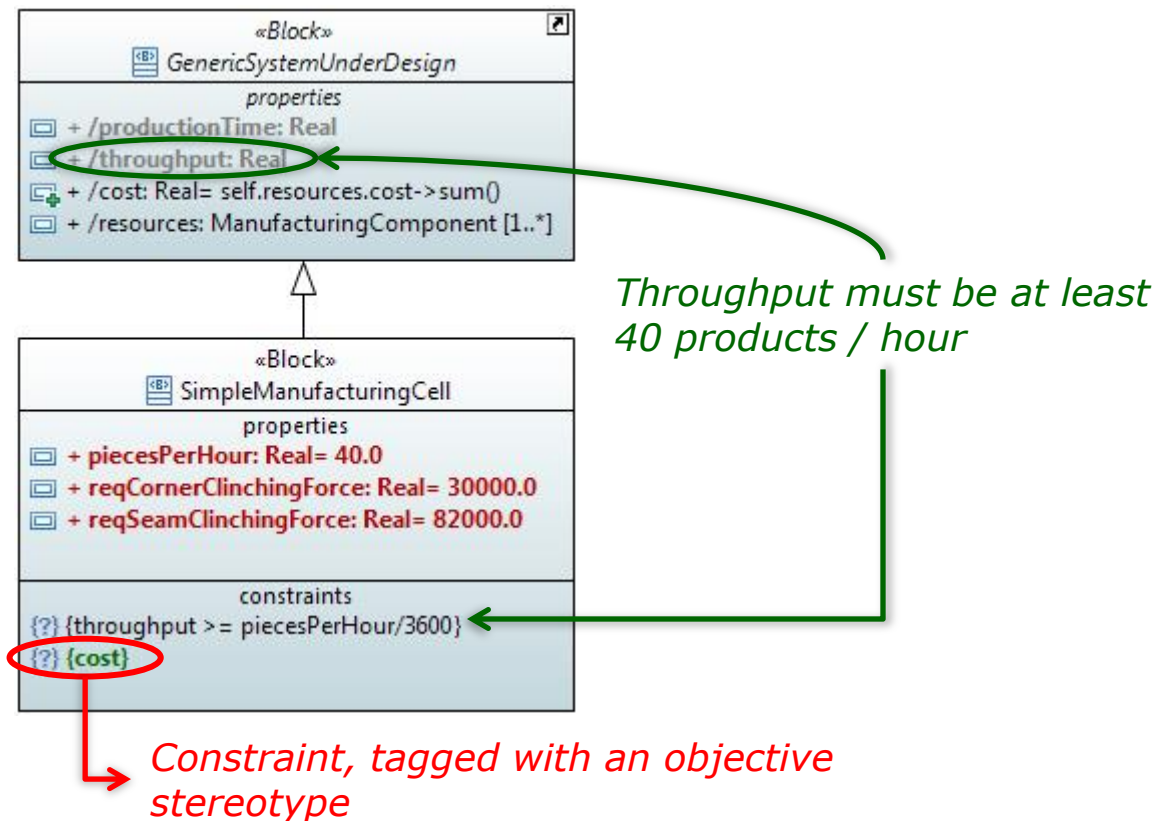
Analysis & Evaluation (MILP)



Design space model

Problem definition

▲ System-Under-Design, Objective & Requirements



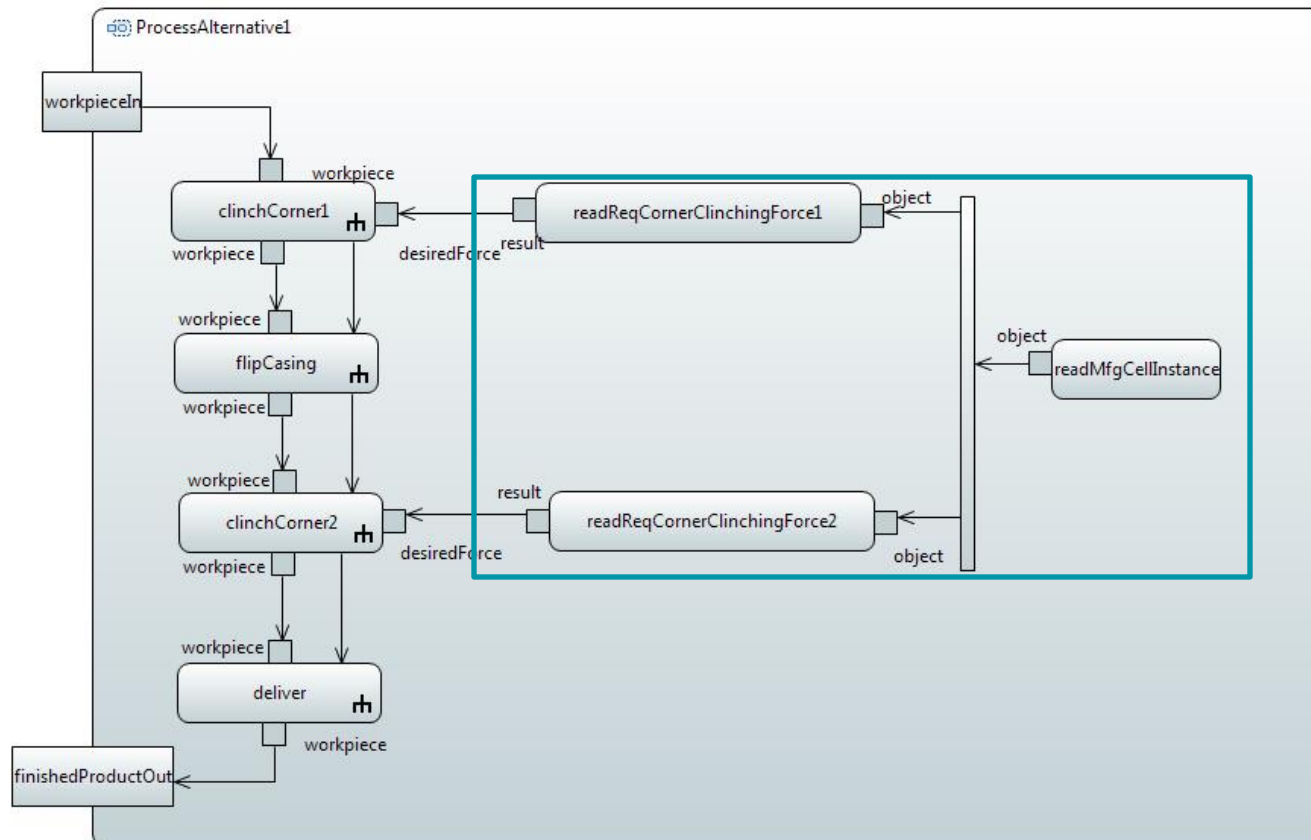
Design space model

Problem definition

Suggestions for more intuitive representations?
- Textual?

▲ Functional specification

▲ Simplified sequential process shown

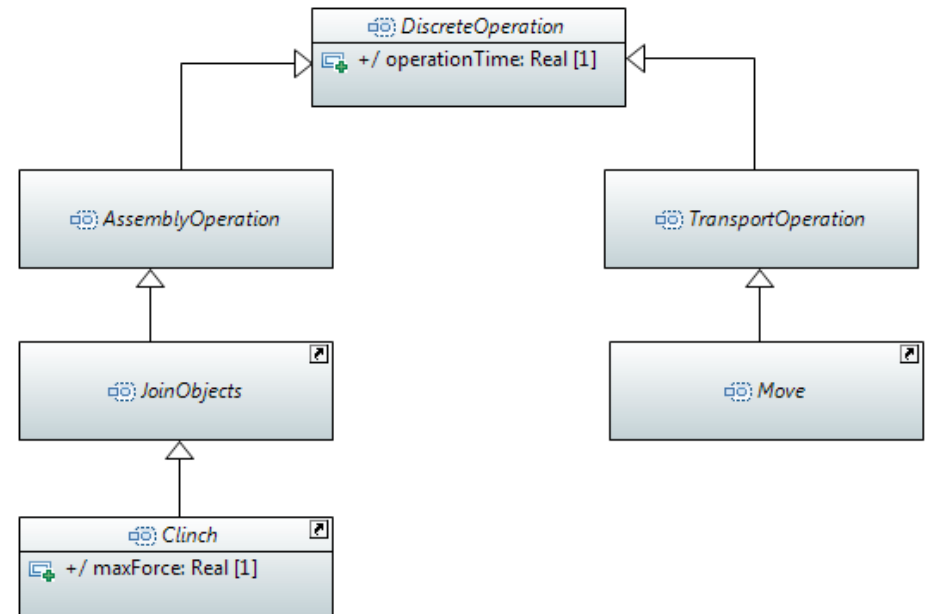


Design space model

Design repository

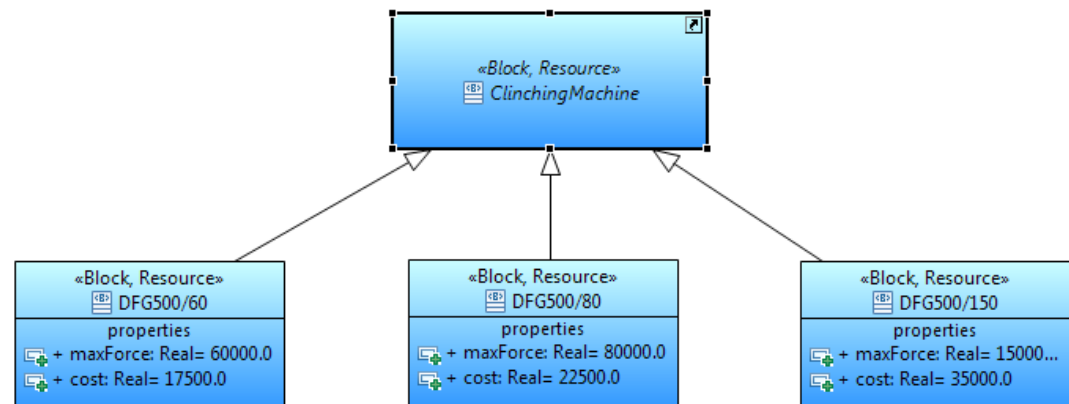
▲ Activities

- ▲ Activity hierarchy
- ▲ Assumes property and parameter equivalence
 - Inheritance
 - Comparison through OCL constraints
 - Visualization in class diagram



▲ Resources

- ▲ Resource hierarchy
- ▲ Non-abstract leaves redefine all inherited properties and specify their values

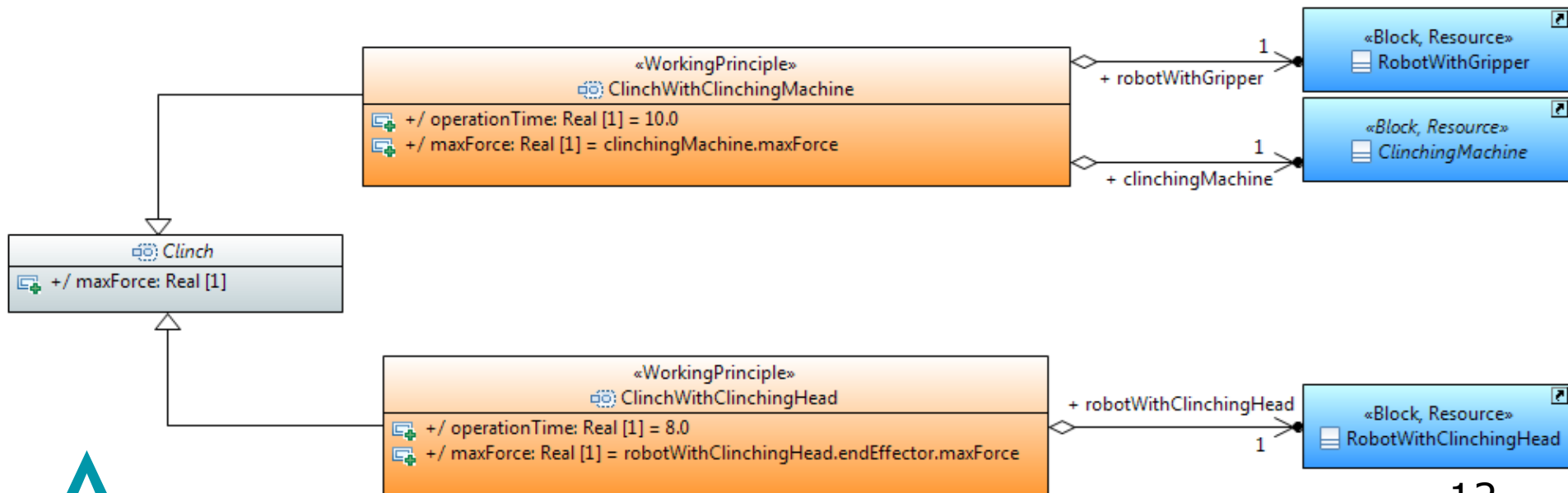


Design space model

Design repository

▲ Working Principles

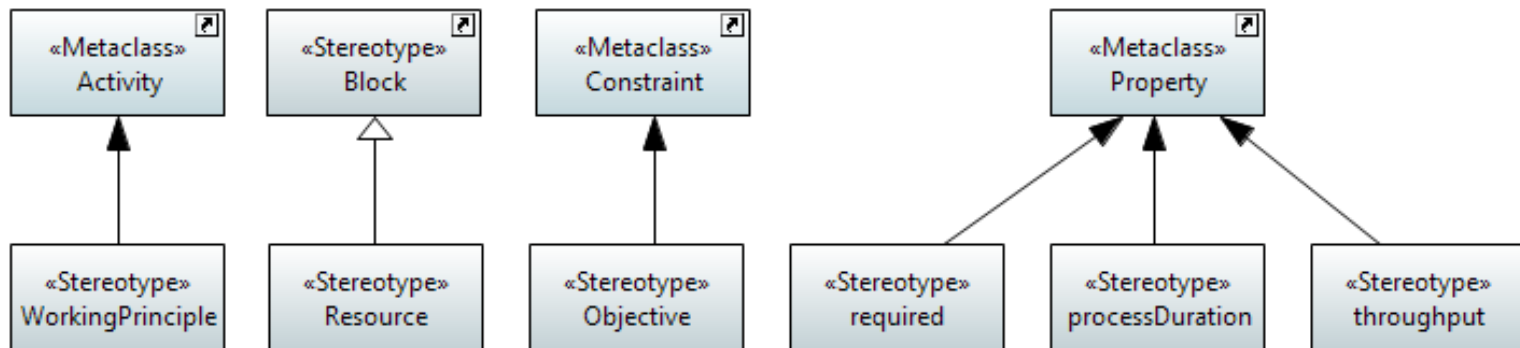
- ▲ Non-abstract leaves in the activity hierarchy
- ▲ Links to the resources needed to execute the activity
- ▲ Redefines properties and fills in their value, often based on resources used



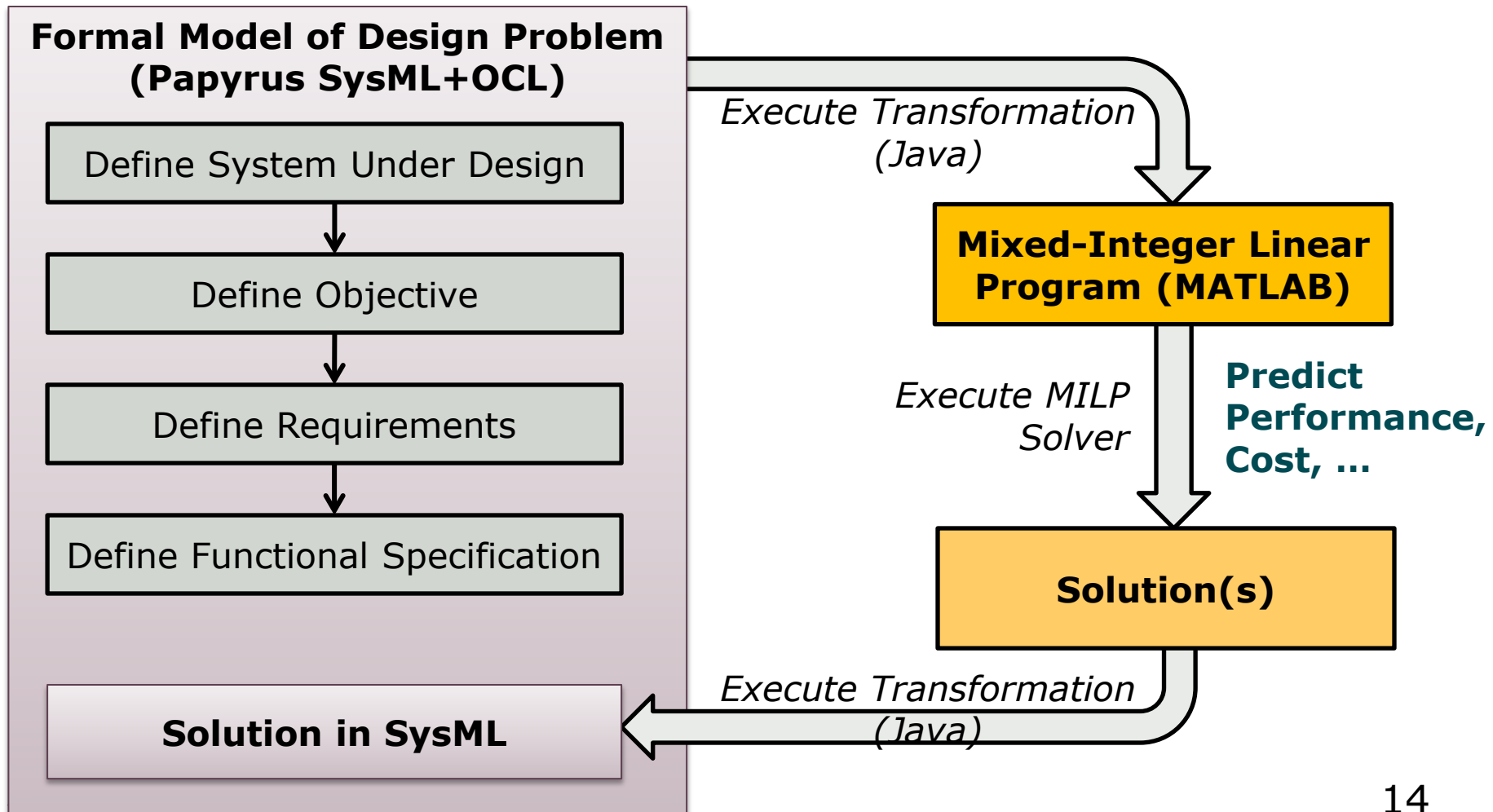
Design space model

Specialized profile

- ▲ Extensive use of existing concepts
 - ▲ Inheritance, redefinition (not present in ecore), derived union,...
- ▲ Limited set of problem-specific concepts needed
 - ▲ Some only needed because of validation



Toolchain



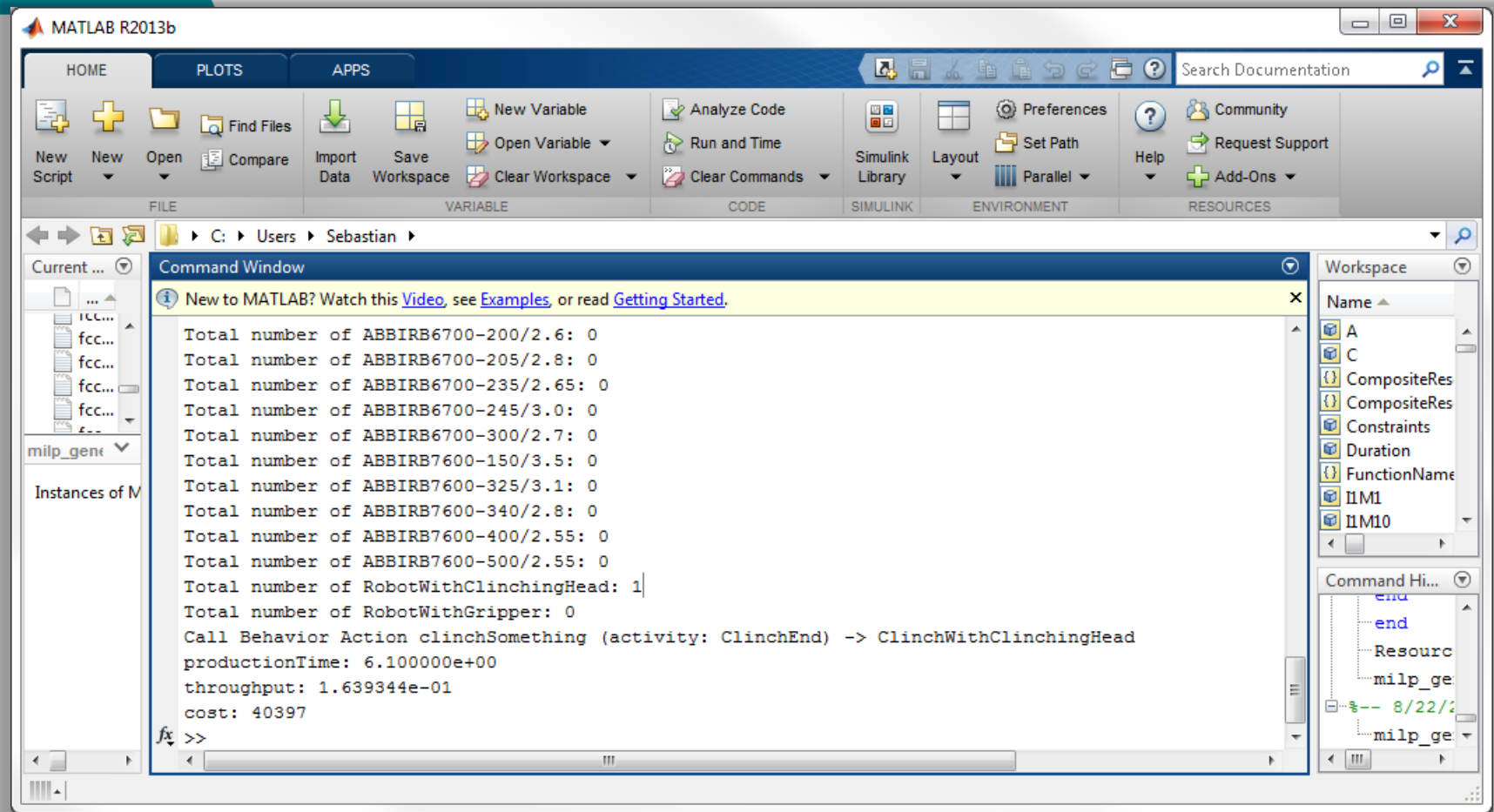
Automated Transformation to MILP & Execution of Solver

The screenshot displays the Eclipse IDE interface with the Papyrus plugin. The top menu bar includes File, Edit, Diagram, Navigate, Search, Papyrus, Project, Run, Transformations, Window, and Help. The 'Transformations' menu is open, showing 'SysML2MILP' and 'Transform and Execute' options. The left sidebar shows a project tree with 'RootElement' expanded, containing '«Block» MySystemUnderDesign', '«Objective» Constraint1', '«Generalization» GenericS', 'MyFunctionalSpecification', '«Block» MyWorkpiece', 'Diagram Problem Definition', and 'Diagram Function Definition'. The main editor window shows the 'milp_generated.m' file, which contains MATLAB code for solving a MILP problem. The code includes variable declarations, constraints, and solver settings. The 'Run' button in the toolbar is highlighted with a green circle.

```
1 - yalmip('clear');
2 - clear all;
3 - clc;
4 -
5 -
6 - A = binvar(40, 2);
7 - assign(A, 0);
8 - C = binvar(40, 2);
9 - assign(C, 0);
10 - Duration = sdpvar(1); % Duration of complete process (time spent working on input element from start t
11 - Constraints = [binary(A), binary(C)]; % Vector of linear constraints
12 - IM = []; % Vector of all resource instances (or, rather, binary variables indicating their existence)
13 - ResourceTypeList = {}; % (Ordered) list of qualified names of types of resource instances (used for cor
14 - CompositeResourceTypeList = {}; % (Ordered) list of qualified names of types of composite resource insta
15 - WPTYPEList = {}; % (Ordered) list of qualified names of types of working principle (used for correspo
16 - FunctionNameList = {}; % (Ordered) list of names of functions (used for correspondence and visualizatio
17 -
18 - % Instances of MZD25/3
19 - I1M1_type = {'DSELibrary::Resources::Joining::Clinching::ClinchingHeads::MZD25/3'}; % Qualified name of
20 - I1M1 = binvar(1);
21 - I1M1_cost = 3000.0; % cost of MZD25/3
22 - I1M1_maxForce = 30000.0; % maxForce of MZD25/3
```

script Ln 952 Col 1

Generation of Solution Instance(s)



The image shows the MATLAB R2013b interface. The Command Window displays the following output:

```
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

Total number of ABBIRB6700-200/2.6: 0
Total number of ABBIRB6700-205/2.8: 0
Total number of ABBIRB6700-235/2.65: 0
Total number of ABBIRB6700-245/3.0: 0
Total number of ABBIRB6700-300/2.7: 0
Total number of ABBIRB7600-150/3.5: 0
Total number of ABBIRB7600-325/3.1: 0
Total number of ABBIRB7600-340/2.8: 0
Total number of ABBIRB7600-400/2.55: 0
Total number of ABBIRB7600-500/2.55: 0
Total number of RobotWithClinchingHead: 1
Total number of RobotWithGripper: 0
Call Behavior Action clinchSomething (activity: ClinchEnd) -> ClinchWithClinchingHead
productionTime: 6.100000e+00
throughput: 1.639344e-01
cost: 40397

fx >>
```

The Workspace panel on the right shows the following variables:

- A
- C
- CompositeRes
- CompositeRes
- Constraints
- Duration
- FunctionName
- IIM1
- IIM10

The Command History panel shows the following commands:

- end
- end
- Resourc
- milp_ge
- 8/22/2
- milp_ge

Excel Output

Resources

Function

Working Principle

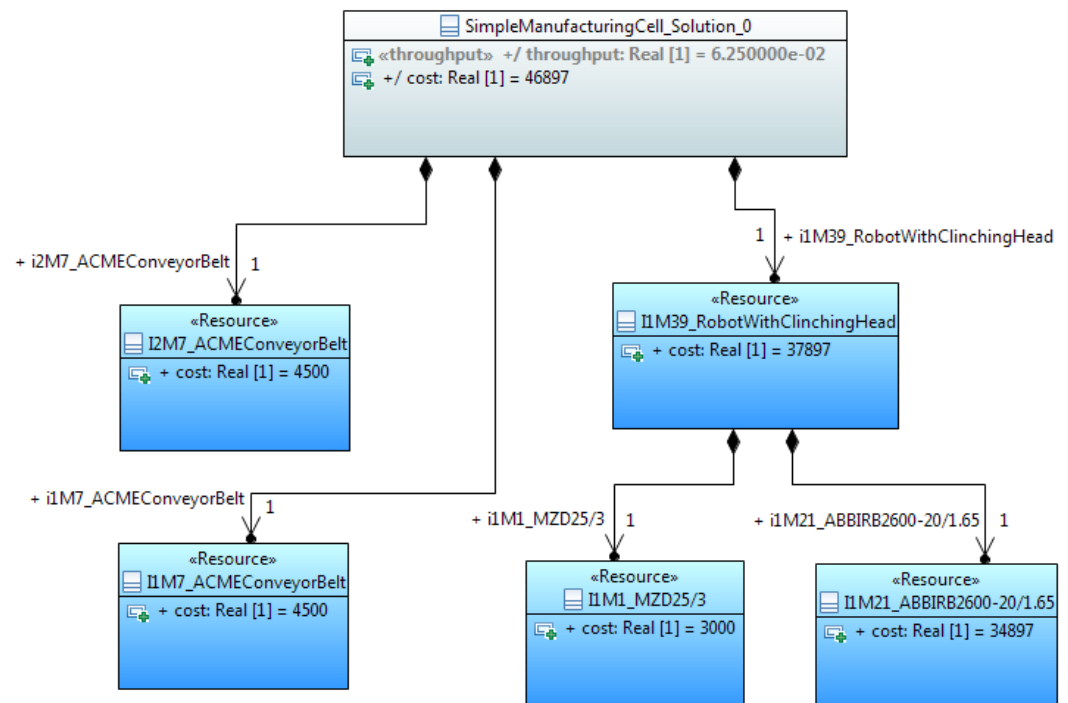
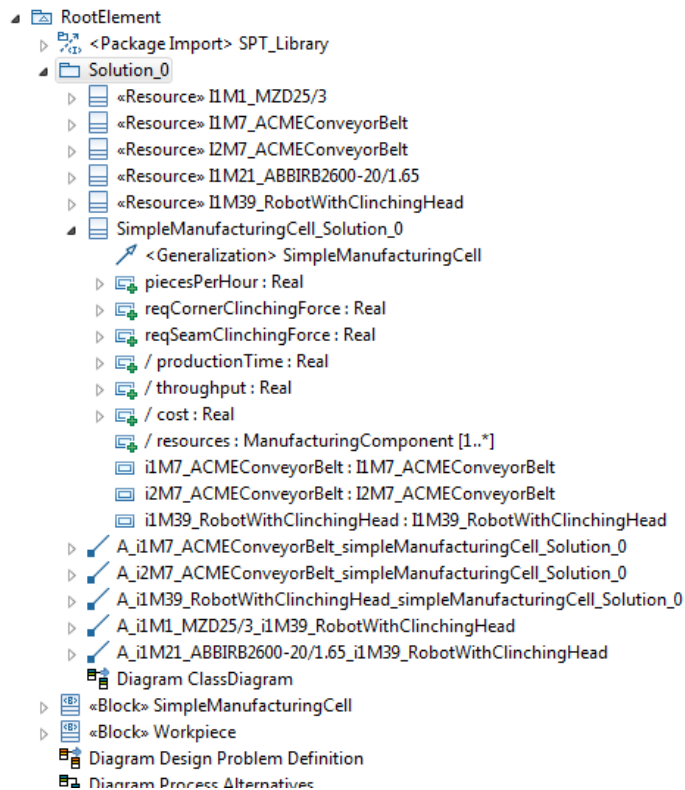
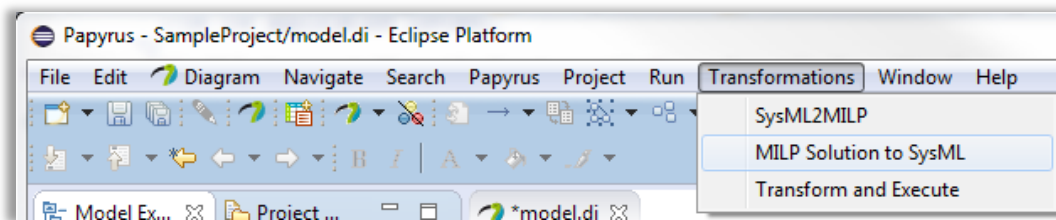
	A	B	C	
1		clinchCorner1 (ClinchWithClinchingHead)	clinchCorner2 (ClinchWithClinchingHead)	fl
2	MZD25/3	1	1	
3	ACMEConveyorBelt	0	0	
4	ABBIRB2600-20/1.65	1	1	
5	RobotWithClinchingHead	1	1	

"1" if function
allocated to resource

	A	B
1		RobotWithClinchingHead
2	MZD25/3	1
3	ABBIRB2600-20/1.65	1
4	RobotWithClinchingHead	1



Back Transformation



Outline

▲ Use case

- ▲ Introduction
- ▲ Design space model
- ▲ Toolchain

▲ Discussion

- ▲ Priorities
- ▲ Tool usability
- ▲ OCL framework
- ▲ Instance creation and visualization

*All of these points have
been added to the bugzilla*



Priorities

0 Governance

- ▲ Clear priorities
- ▲ Bugzilla

1. Usability

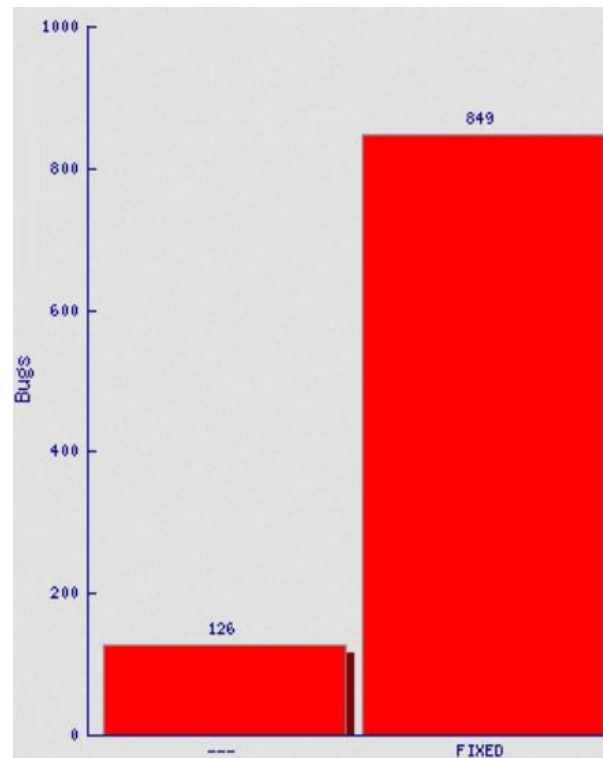
- ▲ Simplification
- ▲ Documentation
- ▲ User-friendliness
- ▲ Customization

2. Robustness

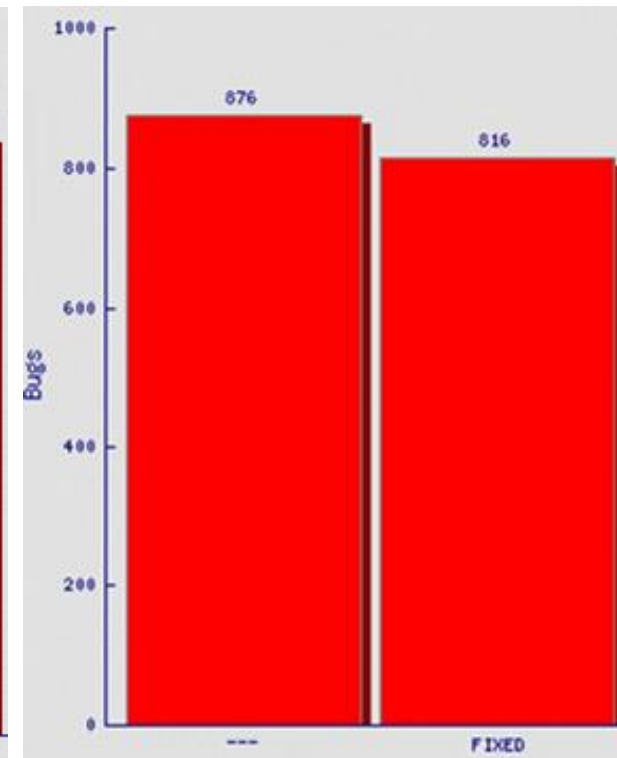
3. Communication

4. Features

2010-2011



01/01/2015-now





Priorities

0 Governance

1. Usability

1. **Simplification**
2. **Documentation**
3. **User-friendliness**
4. **Customization**

You can implement all the features we are asking for, but if the tool is too difficult to use, we still won't be able to get any of our member companies interested

2. Robustness

3. Communication

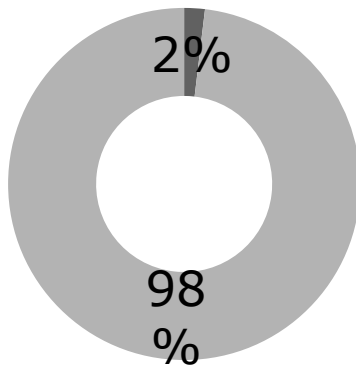
4. Features



Tool usability

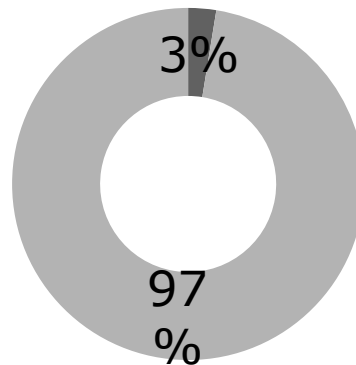
- ▲ Shield the user from the complexities of SysML
 - ▲ Typical DSL only needs a limited set of UML and SysML concepts

**UML
classes**



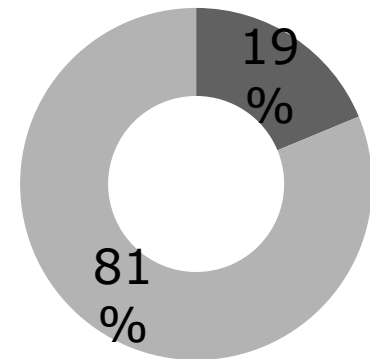
■ Used
■ Unused

**SysML
classes**



■ Used
■ Unused

Diagrams



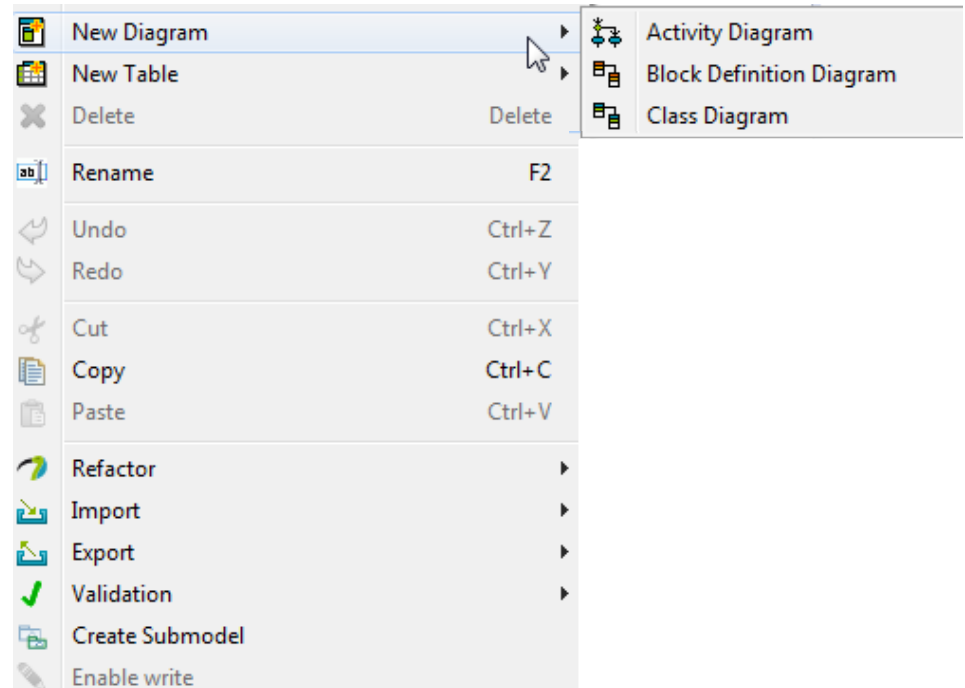
■ Used
■ Unused



Tool usability

- ▲ Filtering of the content the user can use
 - ▲ Limit the number of diagrams/tables
 - ▲ Manipulate the new child menus
 - Single new child menu, containing just the elements we need
 - ▲ Simplification of the palette

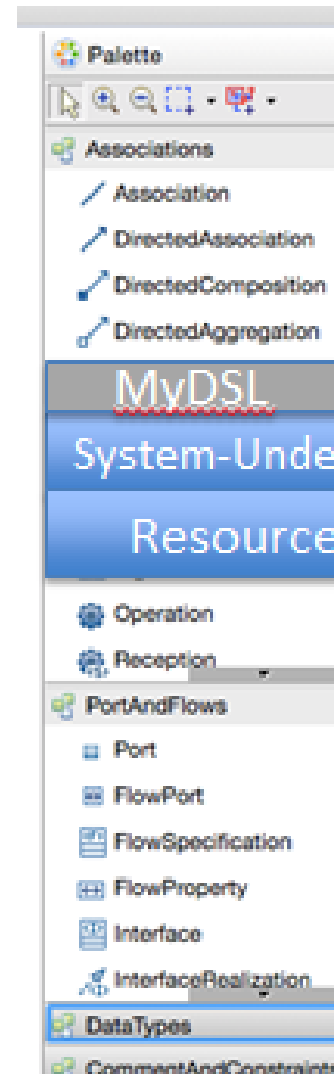
+ documentation of these features



Tool usability

- ▲ Add elements to the palette and menus with more complex functionality
 - ▲ Pre-stereotyped elements
 - ▲ Elements inheriting from a library element
- ▲ Synchronization between palette and new child menu customizations?
 - Currently requires double work

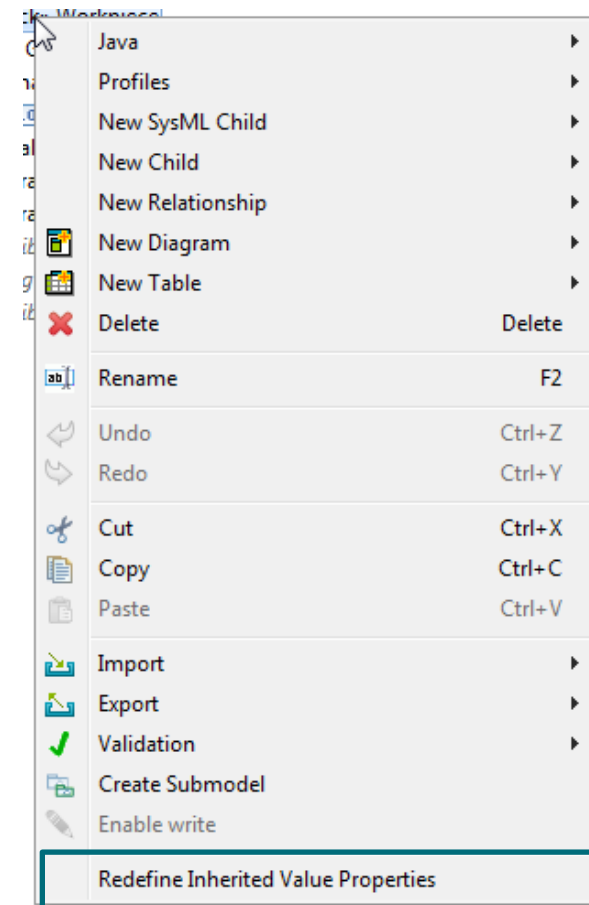
+ documentation of these features



Tool usability

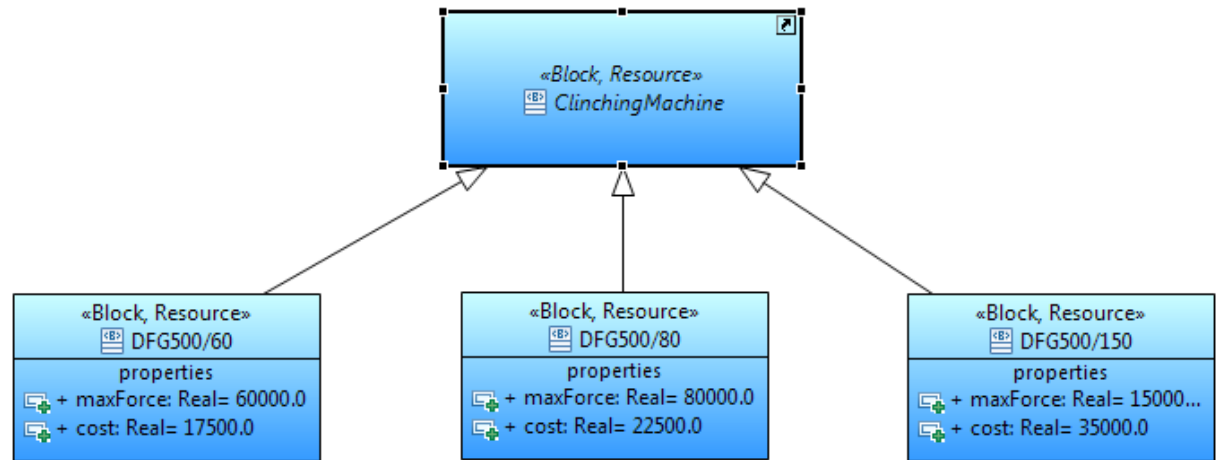
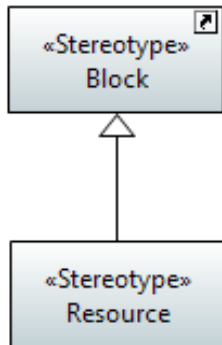
- ▲ Add elements to the palette and menus with more complex functionality
 - ▲ Pre-stereotyped elements
 - ▲ Elements inheriting from a library element
 - ▲ Complex functions to perform common tasks
 - Redefinition of all inherited properties

+ documentation of these features



Tool usability

▲ Additional validation rules



Tool usability

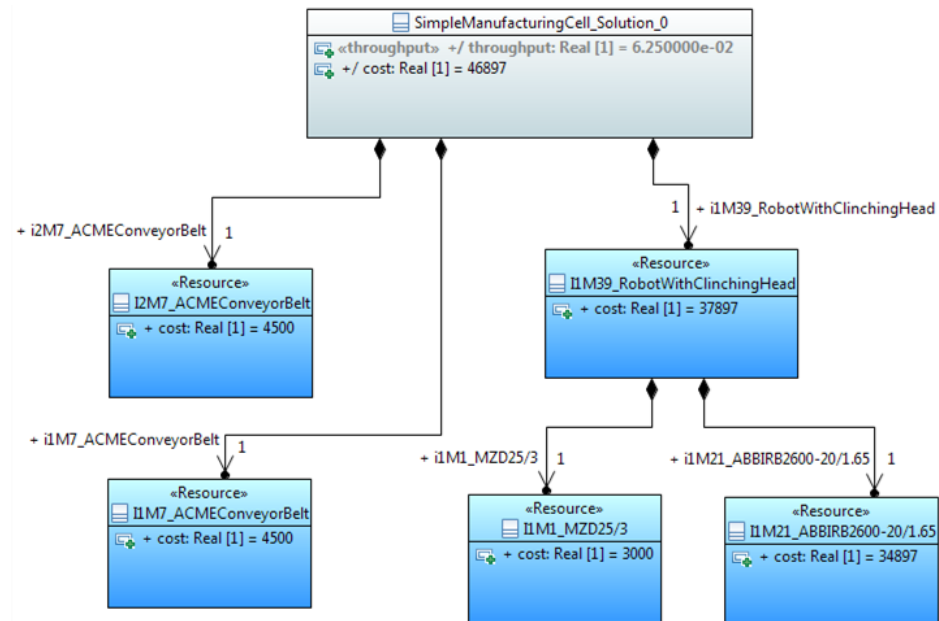
▲ Intelligent lay-outing of diagrams

- ▲ Automatic lay-outing
- ▲ Semantic lay-outing

e.g. Backtransformed solutions

▲ Solution_0

- ▶ «Resource» I1M1_MZD25/3
- ▶ «Resource» I1M7_ACMEConveyorBelt
- ▶ «Resource» I2M7_ACMEConveyorBelt
- ▶ «Resource» I1M21_ABBIRB2600-20/1.65
- ▶ «Resource» I1M39_RobotWithClinchingHead
- ▶ SimpleManufacturingCell_Solution_0
- ▶ A_i1M7_ACMEConveyorBelt_simpleManufacturingCell_Solution_0
- ▶ A_i2M7_ACMEConveyorBelt_simpleManufacturingCell_Solution_0
- ▶ A_i1M39_RobotWithClinchingHead_simpleManufacturingCell_Solution_0
- ▶ A_i1M1_MZD25/3_i1M39_RobotWithClinchingHead
- ▶ A_i1M21_ABBIRB2600-20/1.65_i1M39_RobotWithClinchingHead

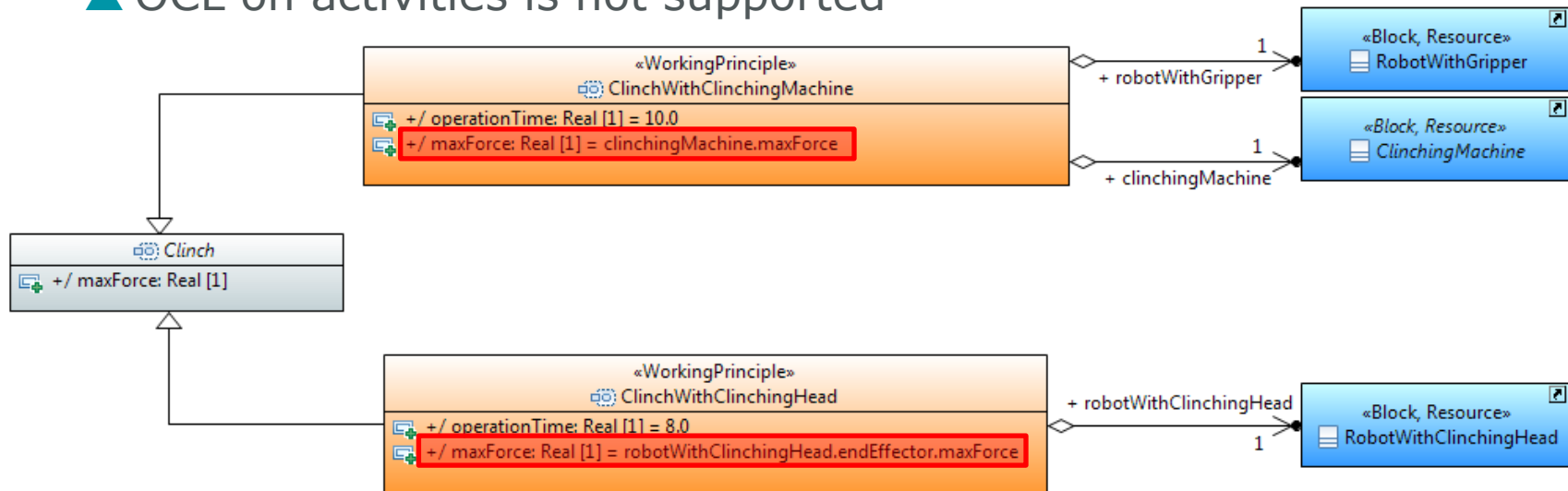


+ documentation of these features



OCCL framework

▲ OCL on activities is not supported

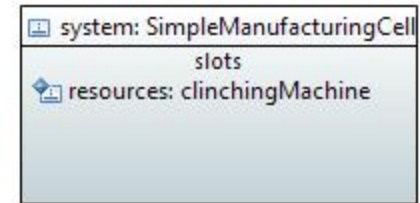
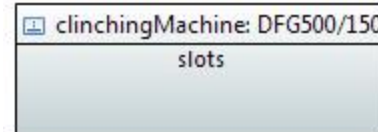


▲ Custom OCL translation

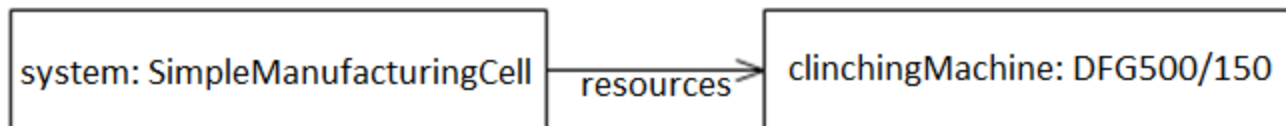
- ▲ Manually created parser (monstrous spaghetti code)
- ▲ A proper implementation would require an OCL metamodel



Instance creation and visualization



- ▲ Instances allow to
 - ▲ Specify partial solutions as 'expert knowledge'
 - ▲ Debugging through validation of correct and incorrect instances
 - ▲ Specification of context specific values
- ▲ Papyrus support for instances is lacking
 - ▲ Instance creation is tedious
 - Wizard, similar to MagicDraw, for creating feasible instances?
 - ▲ No proper visualization provided



Conclusions

- ▲ Papyrus can be used as part of a computational design space exploration toolchain
 - ▲ General purpose languages such as SysML+OCL contain most of the concepts necessary to express the design space
 - ▲ Usability & customizability is a **condicio sine qua non** in order to achieve adoption amongst industrial designers
 - ▲ Design space exploration tools still require interaction with the designer, which requires visualization of solution instances
- ▲ Papyrus seems to be a viable candidate to build upon, yet there remains a lot to do...



**QUESTIONS?
REMARKS?**

Thank you!

